Embedded nonlinear model predictive control for obstacle avoidance using PANOC

Sathya, Ajay
Department of Mechanical Engineering, MECO research group, KU Leuven

Sopasakis, Pantelis
Department of Electrical Engineering (ESAT-STADIUS), KU Leuven

Ruben Van Parys
Dept. of Mechanical Engineering, MECO research group, KU Leuven

Themelis, Andreas
Department of Electrical Engineering (ESAT-STADIUS), KU Leuven

他

出版情報：2018 European Control Conference (ECC). 4, pp.1523-1528, 2018-11-29. IEEE
バージョン：

Embedded nonlinear model predictive control for obstacle avoidance using PANOC

Ajay Sathya, Pantelis Sopasakis, Ruben Van Parys, Andreas Themelis, Goele Pipeleers and Panagiotis Patrinos

***Abstract*— We employ the proximal averaged Newton-type method for optimal control (PANOC) to solve obstacle avoidance problems in real time. We introduce a novel modeling framework for obstacle avoidance which allows us to easily account for generic, possibly nonconvex, obstacles involving polytopes, ellipsoids, semialgebraic sets and generic sets described by a set of nonlinear inequalities. PANOC is particularly well-suited for embedded applications as it involves simple steps, its implementation comes with a low memory footprint and its fast convergence meets the tight runtime requirements of fast dynamical systems one encounters in modern mechatronics and robotics. The proposed obstacle avoidance scheme is tested on a lab-scale autonomous vehicle.**

***Index Terms*— Embedded optimization, Nonlinear model predictive control, Obstacle avoidance.**

I. Introduction

A. Background and Contributions

Autonomous navigation in obstructed environments is a key element in emerging applications such as driverless cars, fleets of automated vehicles in warehouses and aerial robots performing search-and-rescue expeditions. Well-known approaches for finding collision-free motion trajectories are graph-search methods [1], virtual potential field methods [2] or methods using the concept of velocity obstacles [3].

Recent motion planning research focuses on optimization-based strategies. Here, an optimal motion trajectory is sought while collision-avoidance requirements are imposed as constraints thereon. There is a broad range of such formulations. The most straightforward approach constrains the Euclidean distance between the vehicle and obstacle centers [4]. This, however, only allows to separate spheres and ellipsoids as the computation of distances from arbitrary sets is generally not an easy operation.

Other approaches demand the existence of a hyperplane between the vehicle and obstacle at each time instant [5], [6]. This allows the separation of general convex sets. Some methods formulate the collision avoidance requirement by mixed-integer constraints [7]. These types of problems are, however, cumbersome to solve in real-time. For some specific problem types it is possible to transform the system dynamics such that collision-avoidance translates into simple box constraints on the states [8], [9].

Model predictive control (MPC) is a powerful control strategy where control actions are computed by optimizing a cost function which is chosen in order to achieve a control task. Constraints on states, inputs and outputs can be seamlessly incorporated into such a framework. When the system dynamics is linear, the constraints are affine and the cost functions are quadratic, the associated optimization problem is a quadratic program. There exists a mature machinery of convex optimization algorithms [10], [11] which are fast, robust and possess global convergence guarantees which can be used to solve these problems.

Nevertheless, the dynamics of most systems of interest are better modeled by nonlinear equations and constraints are often nonconvex. This situation is very common in obstacle avoidance involving nonconvex optimization problems. These are commonly solved using *sequential quadratic programming* (SQP) [10] and *interior point* (IP) methods [12] which are not well-suited for embedded applications with tight runtime requirements. Nonlinear MPC is often performed using the real-time iteration scheme proposed in [13] which trades speed for accuracy and is accompanied by global convergence guarantees under certain assumptions.

For an algorithm to be suitable for an embedded implementation, it needs to involve only simple steps. This deems methods of the forward-backward-splitting (FBS) type [14], such as the proximal gradient method [15, Sec. 2.3], appealing candidates. FBS-type algorithms can be used to solve nonlinear optimal control problems with simple input constraints via first eliminating the state sequence and expressing the cost as a function of the sequence of inputs alone — the so-called *single shooting* formulation. However, despite its simplicity, FBS, like all first-order methods, can exhibit slow convergence. Its convergence rate is at best Q-linear with a Q-factor close to one for ill-conditioned problems such as most nonlinear MPC problems.

In this work we propose a new modeling framework for generic constraints which can accommodate general nonconvex sets. The proposed methodology assumes that the obstacles are described by a set of nonlinear inequalities and does not require the computation of projections or distances to them. Then, the obstacle avoidance constraints are written as a nonlinear equality constraint involving a smooth function which, in turn, is relaxed using a penalty function.

The resulting problems are solved using PANOC, a proximal averaged Newton-type method for optimal control, which was recently proposed in [16]. Gradients of the

[1]A. Sathya, R. Van Parys and G. Pipeleers are with the Dept. of Mechanical Engineering, MECO research group, KU Leuven, 3001 Leuven, Belgium.

[2]P. Sopasakis, A. Themelis and P. Patrinos are with the Dept. of Electrical Engineering (ESAT-STADIUS), KU Leuven, 3001 Leuven, Belgium.

The work of the these authors was supported by the Research Foundation Flanders (FWO) PhD fellowship 1196818N; FWO research projects G086518N and G086318N; KU Leuven internal funding StG/15/043; Fonds de la Recherche Scientifique – FNRS and the Fonds Wetenschappelijk Onderzoek – Vlaanderen under EOS Project no 30468160 (SeLMA). R. Van Parys is a PhD fellow of the Research Foundation Flanders (FWO-Flanders).

cost function can be efficiently computed using automatic differentiation toolboxes such as CasADi. The algorithm is simple to implement, yet robust, since it combines projected gradient iterations with quasi-Newtonian directions to achieve fast convergence.

The proposed framework is tested on a number of simulation scenarios where we show that it is possible to avoid obstacles of complex shape described by nonlinear inequalities. PANOC is compared with SQP and IP methods and is found to be significantly faster. Furthermore, we present experimental results on a lab-scale robotic platform which runs a C implementation of PANOC.

B. Notation

Let $\mathbb{N}$ be the set of nonnegative integers, $\mathbb{N}_{[k_1,k_2]}$ be the set of integers in the interval $[k_1, k_2]$ and $\overline{\mathbb{R}} = \mathbb{R} \cup \{+\infty\}$ be the set of extended real numbers. For a matrix $A \in \mathbb{R}^{m\times n}$, we denote its transpose by $A^\top$. For $x \in \mathbb{R}$, we define the operator $[x]_+ = \mathbf{max}\{x, 0\}$. For a nonempty closed convex set $U \subseteq \mathbb{R}^n$, the *projection* onto U is the operator $\mathbf{\Pi}_U(v) = \mathbf{argmin}_{u\in U} \|u - v\|$. The *distance* from the set U is defined as $\mathbf{dist}_U(v) = \mathbf{inf}_{u\in U} \|u - v\|$. The class of continuously differentiable functions $f : \mathbb{R}^n \to \mathbb{R}$ is denoted by C^1. The subset of C^1 of functions with Lipschitz-continuous gradient is denoted as $C^{1,1}$. We use the notation $C^{1,1}_L$ for $C^{1,1}$ functions with L-Lipschitz gradients.

II. NMPC for obstacle avoidance

A. Problem statement

Kinematic equations lead to continuous-time nonlinear dynamical systems of the form $\dot{x} = f_c(x, u, t)$ where $x \in \mathbb{R}^{n_x}$ is the system state, typically a vector comprising of position, velocity and orientation data, and $u \in \mathbb{R}^{n_u}$ is the control signal. We assume that the position coordinates $z \in \mathbb{R}^{n_d}$ are part of the state vector. The continuous-time dynamics can be discretized (for instance, using an explicit Runge-Kutta method) leading to a discrete-time dynamical system of the form

$$x_{k+1} = f_k(x_k, u_k). \tag{1}$$

As it is typically the case in practice, we assume that $f_k : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ are smooth mappings.

The objective of the navigation controller is to steer the controlled vehicle from an initial state x_0 to a target state x_{ref}, typically a position in space together with a desired orientation. At the same time, the vehicle has to avoid certain, possibly moving, obstacles which are described by open sets $O_{kj} \subseteq \mathbb{R}^{n_d}$, $j \in \mathbb{N}_{[1,q_k]}$, each described by

$$O_{kj} = \{z \in \mathbb{R}^{n_d} : h^i_{kj}(z) > 0, i \in \mathbb{N}_{[1,m_{kj}]}\}. \tag{2}$$

Sets O_{kj} need not be convex. Obstacle avoidance constraints can be concisely written as

$$z_k \notin O_{kj}, \text{ for } j \in \mathbb{N}_{[1,q_k]}. \tag{3}$$

Moreover, the vehicle is only allowed to move in a domain which is described by the inequalities

$$g_k(x_k, u_k) \le 0, \tag{4}$$

where $g_k : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_c}$ is a C^2 mapping and $\le$ is meant in the element-wise sense. Control actions u_k are constrained in a closed compact set U_k, that is

$$u_k \in U_k, \tag{5}$$

on which it is easy to project and hereafter shall be assumed to be convex. Sets U_k often represent box constraints of the form $U_k = \{u \in \mathbb{R}^{n_u} : u_{\min} \le u \le u_{\max}\}$.

B. Nonlinear model predictive control

Nonlinear model predictive control problems arising in obstacle avoidance can be written in the following form

$$\begin{aligned}
\textbf{minimize} \quad & \ell_N(x_N) + \sum_{k=0}^{N-1} \ell_k(x_k, u_k), && \text{(6a)}\\
\textbf{subject to} \quad & x_0 = x, && \text{(6b)}\\
& x_{k+1} = f_k(x_k, u_k),\ k \in \mathbb{N}_{[0,N-1]}, && \text{(6c)}\\
& u_k \in U_k,\ k \in \mathbb{N}_{[0,N-1]}, && \text{(6d)}\\
& z_k \notin O_{kj},\ j \in \mathbb{N}_{[1,q_k]},\ k \in \mathbb{N}_{[0,N]} && \text{(6e)}\\
& g_k(x_k, u_k) \le 0,\ k \in \mathbb{N}_{[0,N]}, && \text{(6f)}\\
& g_N(x_N) \le 0. && \text{(6g)}
\end{aligned}$$

The stage costs $\ell_k : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$ for $k \in \mathbb{N}_{[0,N-1]}$ in (6a) are $C^{1,1}$ functions penalizing deviations of the state from the reference (destination and orientation) and may be taken to be quadratic functions of the form $\ell(x_k, u_k) = (x_k - x_{\mathrm{ref}})^\top Q_k(x_k - x_{\mathrm{ref}}) + (u_k - u_{\mathrm{ref}})^\top R_k(u_k - u_{\mathrm{ref}})$. The terminal cost $\ell_N : \mathbb{R}^{n_x} \to \mathbb{R}$ in (6a) is a $C^{1,1}$ function such as $\ell_N(x_N) = (x_N - x_{\mathrm{ref}})^\top Q_N(x_N - x_{\mathrm{ref}})$.

C. Reformulation of obstacle avoidance constraints

Consider an obstacle described as the intersection of a finite number of strict nonlinear inequalities

$$O = \{z \in \mathbb{R}^{n_d} : h^i(z) > 0, i \in \mathbb{N}_{[1,m]}\}, \tag{7}$$

where $h^i : \mathbb{R}^{n_d} \to \mathbb{R}$ are $C^{1,1}$ functions. The constraint $z \notin O$ — cf. (6e) — is satisfied if and only if

$$h^{i_0}(z) \le 0, \text{ for some } i_0 \in \mathbb{N}_{[1,m]}, \tag{8}$$

or, equivalently, $\left[h^{i_0}(z)\right]^2_+ = 0$. This constraint can then be encoded as

$$\psi_O(z) := \tfrac{1}{2} \prod_{i=1}^{m} \left[h^i(z)\right]^2_+ = 0. \tag{9}$$

We have expressed the obstacle avoidance constraints as a nonlinear equality constraint. We should remark that, unlike approaches based on the distance-to-set function [17], function ψ_O in (9) is a C^1 function of z. Indeed, ψ_O is differentiable in $\mathbb{R}^{n_d} \setminus O$ (and equal to 0), it is differentiable in O with gradient

$$\nabla\psi_O(z) = \begin{cases} \sum_{i=1}^{m} h^i(z) \prod_{j\neq i} (h^j(z))^2 \nabla h^i(z), & \text{if } x \in O, \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

Note that $\nabla\psi_O$ on the boundary of O vanishes, so it is everywhere continuous. If, additionally, O is bounded, ψ_O is $C^{1,1}$.

The formulation of Eq. (9) can be used for obstacles described by quadratic constraints of the form $O = \{z \in \mathbb{R}^{n_d} : 1 - (z-c)^\top E(z-c) > 0\}$, such as balls and ellipsoids. Then, the associated equality constraint becomes $[1-(z-c)^\top E(z-c)]_+^2 = 0$. Polyhedral obstacles of the form $O = \{z \in \mathbb{R}^{n_d} : b_i - a_i^\top z > 0, \in \mathbb{N}_{[1,m]}\}$, with $b_i \in \mathbb{R}$ and $a_i \in \mathbb{R}^{n_d}$, can also be accommodated by the constraint $\prod_{i=1}^m [b_i - a_i^\top z]_+^2 = 0$.

Equation (9) can also be used to describe general nonconvex constraints such as ones described by semi-algebraic sets where $h^i(z)$ are polynomials as well as any other obstacle which is available in the aforementioned representation.

Obstacle avoidance constraints (9) are equivalent to the existence of $t \in \mathbb{R}^m$ so that

$$\mathbf{min}\{t_1, \dots, t_m\} = 0, \tag{11a}$$
$$h^i(z) \le t_i, \text{ for } i \in \mathbb{N}_{[1,m]}. \tag{11b}$$

This observation reveals a link to vertical complementarity constraints which have been studied extensively in the literature [18], [19].

D. Relaxation of constraints

Due to the fact that modeling errors and disturbances may lead to the violation of imposed constraints and infeasibility of the MPC optimization problem, it is common practice in MPC to replace state constraints by appropriate penalty functions known as *soft constraints*. Quadratic penalty functions are often used for this purposes revealing a clear link between this approach and the *quadratic penalty method* in numerical optimization [15, Sec. 4.2.1].

Equality constraints, such as the ones arising in the reformulation of the obstacle avoidance constraints in Section II-C, can be relaxed by means of soft constraints. Indeed, constraints of the form $\Phi_k(z_k) = 0$, $k \in \mathbb{N}_{[1,N]}$, where $\Phi_k : \mathbb{R}^{n_d} \to \mathbb{R}_+$ are C^2 functions, can be relaxed by introducing the penalty functions $\tilde{\Phi}_k(z_k) = \eta_k \Phi_k(z_k)$ for some weight factors $\eta_k > 0$.

That said, constraints like (9) for a set of time-varying obstacles $O_{kj} = \{z \in \mathbb{R}^{n_d} : h_{kj}^i(z) > 0, i \in \mathbb{N}_{[1,m_{kj}]}\}$, with $j \in \mathbb{N}_{[1,q_k]}$, can be relaxed by the appending the following term in the original cost function

$$\tilde{h}_k(z_k) = \sum_{j=1}^{q_k} \eta_{kj} \prod_{i=1}^{m_{kj}} \left[h_{kj}^i(z)\right]_+^2, \tag{12}$$

for some positive weight factors $\eta_{kj} > 0$.

Note that the proposed approach for dealing with obstacle avoidance constraints requires only a representation of the obstacles in the generic form (7) and does not call for the computation of distances to the obstacles as in distance-based methods [4], [17], nor does it require the obstacles to be convex sets.

Similarly, inequality constraints of the form $g_k(x_k, u_k) \le 0$ and $g_N(x_N) \le 0$ can be relaxed by introducing the penalty functions $\tilde{g}_k(x,u) = \beta_k [g_k(x,u)]_+^2$ and $\tilde{g}_N(x) = \beta_N [g_N(x)]_+^2$ for positive weights $\beta_k > 0$, $k \in \mathbb{N}_{[0,N]}$.

We may now relax the state constraints in (6) by defining the modified stage cost and terminal cost functions

$$\tilde{\ell}_k(x,u) = \ell_k(x,u) + \tilde{g}_k(x,u) + \tilde{h}_k(z_k),$$
$$\tilde{\ell}_N(x) = \ell_N(x) + \tilde{g}_N(x),$$

leading to the following relaxed optimization problem without state constraints

$$\textbf{minimize} \quad \tilde{\ell}_N(x_N) + \sum_{k=0}^{N-1} \tilde{\ell}_k(x_k, u_k), \tag{13a}$$
$$\textbf{subject to} \quad x_0 = x, \tag{13b}$$
$$x_{k+1} = f_k(x_k, u_k),\ k \in \mathbb{N}_{[0,N-1]}, \tag{13c}$$
$$u_k \in U_k,\ k \in \mathbb{N}_{[0,N-1]}, \tag{13d}$$

E. NMPC problem formulation

In this section we cast the nonlinear MPC problem (6) as

$$\underset{u \in U}{\textbf{minimize}}\ \ell(u), \tag{14}$$

where the optimization is carried out over vectors $u = (u_0, \dots, u_{N-1}) \in \mathbb{R}^n$, with $n = Nu_u$ and $U := U_0 \times U_1 \times \cdots \times U_{N-1}$ and $\ell : \mathbb{R}^n \to \mathbb{R}$ is a real-valued $C_{L_\ell}^{1,1}$ function.

We introduce the following sequence of functions $F_k : \mathbb{R}^n \to \mathbb{R}^{n_x}$ for $k \in \mathbb{N}_{[0,N-1]}$

$$F_0(u) = x, \tag{15a}$$
$$F_{k+1}(u) = f_k(F_k(u), u_k). \tag{15b}$$

Define the smooth function $\ell : \mathbb{R}^n \to \mathbb{R}$

$$\ell(u) := \tilde{\ell}_N(F_N(u)) + \sum_{k=0}^{N-1} \tilde{\ell}_k(F_k(u), u_k). \tag{16}$$

The gradient of function ℓ in (16) can be computed by means of the reverse mode of automatic differentiation (also known as adjoint method or backpropagation) as shown in Alg. 1 [20].

Algorithm 1 Automatic differentiation for ℓ in (16)

Input: $x_0 \in \mathbb{R}^{n_x}$, $u \in \mathbb{R}^n$.
Output: $\ell(u)$, $\nabla\ell(u)$

$\ell(u) \leftarrow 0$
for $k = 0, \dots, N-1$ **do**
  $x_{k+1} \leftarrow f_k(x_k, u_k)$, $\ell(u) \leftarrow \ell(u) + \tilde{\ell}_k(x_k, u_k)$
$\ell(u) \leftarrow \ell(u) + \tilde{\ell}_N(x_N)$, $p_N \leftarrow \nabla\tilde{\ell}_N(x_N)$
for $k = N-1, \dots, 0$ **do**
  $p_k \leftarrow \nabla_{x_k} f_k(x_k, u_k) p_{k+1} + \nabla_{x_k}\tilde{\ell}_k(x_k, u_k)$
  $\nabla_{u_k}\ell_k(u) \leftarrow \nabla_{u_k} f_k(x_k, u_k) + \nabla_{u_k}\tilde{\ell}_k(x_k, u_k)$

Problem (14) is in a form that allows the application of the projected gradient iteration

$$u^{\nu+1} = T_\gamma(u^\nu) := \mathbf{\Pi}_U(u^\nu - \gamma\nabla\ell(u^\nu)), \tag{17}$$

with $\gamma > 0$. In particular, if $\ell \in C_{L_\ell}^{1,1}$ and $\gamma < 2/L_\ell$, then all accumulation points of (17), $u^\star$, are fixed points of T_γ called γ-*critical points*, that is [10, Prop. 2.3.2]

$$u^\star = T_\gamma(u^\star). \tag{18}$$

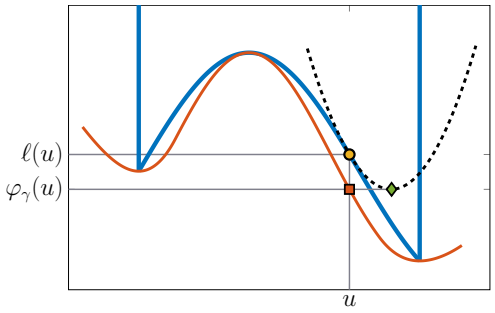

Fig. 1. Construction of the FBE with $\gamma = 0.15$ (red line) for the function $\ell(u) = \mathbf{sin}(2u)$ over the set $U = [0, 2]$ (thick blue line). The dashed line shows the approximation of ℓ at a point u by the quadratic model $Q^\ell_\gamma(v; u)$. The value of FBE at u is then given by $\varphi_\gamma(u) = \mathbf{inf}_{v\in U} Q^\ell_\gamma(v; u)$. Note that the local minima of ℓ over U are exactly the local minima of φ_γ.

III. Fast nonlinear MPC

The problem of finding a fixed point for T_γ can be reduced to the equivalent problem of finding a zero of the *fixed-point residual* operator which is defined as the operator

$$R_\gamma(u) = \tfrac{1}{\gamma}(u - T_\gamma(u)). \tag{19}$$

This motivates the adoption of a Newton-type iterative scheme

$$u^{\nu+1} = u^\nu - H_\nu R_\gamma(u^\nu), \tag{20}$$

where H_ν are invertible linear operators, appropriately chosen so as to encode first-order information about R_γ. This is done by enforcing the *inverse secant condition* $s^\nu = H_{\nu+1}y^\nu$, for $s^\nu = u^{\nu+1} - u^\nu$ and $y^\nu = R_\gamma(u^{\nu+1}) - R_\gamma(u^\nu)$ and can be obtained by quasi-Newtonian methods such as the limited-memory BFGS (L-BFGS) method [10] which is free from matrix operations, requires only a limited number of inner products and is suitable for embedded implementations. The main weakness of this approach is that convergence is only guaranteed in a neighborhood of a γ-critical point $u^\star$. We shall describe a *globalization* procedure which hinges on the notion of the *forward-backward envelope* function.

A. The Forward-Backward envelope function

The *forward-backward envelope* (FBE) is an exact, continuous, real-valued merit function for (14) [16], [21]–[24]. Function ℓ can be approximated at a point $u \in U$ by the quadratic upper bound

$$Q^\ell_\gamma(v; u) = \ell(u) + \nabla\ell(u)^\top(v - u) + {}^1\!/_{2\gamma}\|u - v\|^2. \tag{21}$$

The FBE is then defined as

$$\varphi_\gamma(u) = \inf_{v\in U} Q^\ell_\gamma(v; u). \tag{22}$$

This construction is illustrated in Fig. 1. Provided that it is easy to compute the distance to U, the FBE can be easily computed by

$$\varphi_\gamma(u) = \ell(u) - \tfrac{\gamma}{2}\|\nabla\ell(u)\|^2 + \tfrac{1}{2\gamma}\,\mathbf{dist}^2_U(u - \gamma\nabla\ell(u)). \tag{23}$$

Therefore, the computation of the FBE is of the same complexity as that of a forward-backward step.

The FBE possesses several favorable properties, perhaps the most important being that it is real-valued, continuous and for $\gamma \in (0, {}^1\!/_{L_\ell})$ shares the same (local/strong) minima with the original problem (14). This means that (14) is reduced to an unconstrained minimization problem. Moreover, if $\ell \in C^2$, then $\varphi_\gamma \in C^1$ with $\nabla\varphi_\gamma(u) = (I - \gamma\nabla^2\ell(u))R_\gamma(u)$.

B. PANOC Algorithm

We employ the proximal averaged Newton-type method for optimal control, for short PANOC, which was recently proposed in [16]. PANOC performs fast Newton-type updates while an FBE-based linesearch endows it with global convergence properties while it uses the same oracle as the projected gradient method.

Algorithm 2 PANOC algorithm for problem (14)

Input: $\gamma \in (0, {}^1\!/_{L_\ell})$, $L_\ell > 0$, $\sigma \in (0, \frac{\gamma}{2}(1-\gamma L_\ell))$, $u_0 \in \mathbb{R}^n$, $x_0 \in \mathbb{R}^{n_x}$, L-BFGS memory length μ.

1: **for** $\nu = 0, 1, \ldots$ **do**
2: $\quad\bar{u}^\nu \leftarrow \mathbf{\Pi}_U(u^\nu - \gamma\nabla\ell(u^\nu))$
3: $\quad r^\nu \leftarrow \gamma^{-1}(u^\nu - \bar{u}^\nu)$
4: $\quad d^\nu \leftarrow -H_\nu r^\nu$ using L-BFGS
5: $\quad u^{\nu+1} \leftarrow u^\nu - (1-\tau_\nu)\gamma r^\nu + \tau_\nu d^\nu$, where τ_ν is the largest number in $\{{}^1\!/_{2^i} : i \in \mathbb{N}\}$ such that

$$\varphi_\gamma(u^{\nu+1}) \le \varphi_\gamma(u^\nu) - \sigma\|r^\nu\|^2 \tag{24}$$

The iterative scheme, which is presented in Alg. 2, involves the computation of a projected gradient point $\bar{u}^\nu$ in step 2 and an L-BFGS direction d^ν in step 4. L-BFGS obviates the need to store or explicitly update matrices H_ν in (20) by storing a number μ of past values of s^ν and y^ν. The computation of d^ν requires only inner products which amount to a maximum of $4\mu n$ scalar multiplications. In step 5, the iterates are updated using a convex combination of the projected gradient update direction $-\gamma R_\gamma(u^\nu)$ and a fast quasi-Newtonian direction d^ν. The algorithm is terminated when $\|R_\gamma(u^\nu)\|_\infty$ drops below a specified tolerance.

In line 5, the backtracking line search procedure ensures that a sufficient decrease condition is satisfied using the FBE as a merit function. Under mild assumptions, eventually only fast updates are activated and updates reduce to $u^{\nu+1} = u^\nu + d^\nu$. The sequence of fixed-point residuals $\{r^\nu\}_{\nu\in\mathbb{N}}$ converges to 0 square summably, while PANOC produces sequences of iterates, $\{u^\nu\}_{\nu\in\mathbb{N}}$ and $\{\bar{u}^\nu\}_{\nu\in\mathbb{N}}$, whose cluster points are γ-critical points.

In absence of a Lipschitz constant L_ℓ, the algorithm can be initialized with an estimate, e.g., $L^0_\ell = \|\nabla_u\ell(u^0 + \delta u) - \nabla_u\ell(u^0)\|/\|\delta u\|$, where $\delta u \in \mathbb{R}^n$ is a small perturbation vector. Then, step 2 in Alg. 2 needs to be replaced by the backtracking procedure of Alg. 3 which updates L_ℓ, σ and γ.

Algorithm 3 Lipschitz constant backtracking

while $\ell(\bar{u}^\nu) > \ell(u^\nu) - \gamma\nabla\ell(u^\nu)^\top r^\nu + {}^{L_\ell}\!/_2\|\gamma r^\nu\|^2$ **do**
$\quad L_\ell \leftarrow 2L_\ell$, $\sigma \leftarrow {}^\sigma\!/_2$, $\gamma \leftarrow {}^\gamma\!/_2$
$\quad\bar{u}^\nu \leftarrow \mathbf{\Pi}_U(u^\nu - \gamma\nabla\ell(u^\nu))$

Alg. 3 updates L_ℓ, σ and γ only a finitely many times, so, it does not affect the convergence properties of the algorithm.

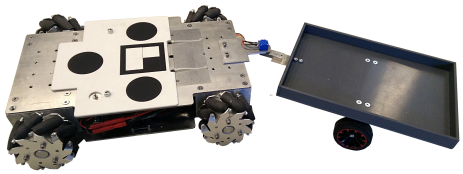

Fig. 2. In-house developed mobile robot with trailer.

In MPC, we may warm start the algorithm using the optimal solution of the previous MPC instance. Unlike SQP and IP methods, PANOC does not require the solution of linear systems or quadratic programming problems at every iteration and it involves only very simple operations such as vector additions, scalar and inner products. Additionally, PANOC converges *globally*, that is, from any initial point $u^0 \in \mathbb{R}^n$.

IV. Simulations

The proposed methodology is validated on the control of a mobile robot carrying a trailer shown in Fig. 2. Assuming zero slip of the trailer wheels, the nonlinear kinematics is

$$\dot{p}_x = u_x + L\,\mathbf{sin}\,\theta \cdot \dot{\theta}, \tag{25a}$$
$$\dot{p}_y = u_y - L\,\mathbf{cos}\,\theta \cdot \dot{\theta}, \tag{25b}$$
$$\dot{\theta} = \tfrac{1}{L}\left(u_y\,\mathbf{cos}\,\theta - u_x\,\mathbf{sin}\,\theta\right), \tag{25c}$$

where the state vector $x = (p_x, p_y, \theta)$ comprises the coordinates p_x and p_y of the trailer and the heading angle θ. The input $u = (u_x, u_y)$ is a velocity reference which is tracked by a low-level controller. The distance between the center of mass of the trailer and the fulcrum connecting to the towing vehicle is $L = 0.5\,\mathrm{m}$. In Fig. 3 we present four obstacle avoidance scenarios involving, among other, obstacles described by polynomial and trigonometric functions. The system dynamics given in (25) is discretized using the fourth-order Runge-Kutta method and we use memory $\mu = 10$ for the L-BFGS directions in Alg. 2.

The single shooting formulation of Section II is solved with PANOC, the interior point solver IPOPT, the forward-backward splitting (FBS) implementation of ForBES and the SQP of MATLAB's `fmincon`. The problem was also brought in a multiple shooting formulation where obstacle avoidance constraints were imposed as in (9) and it was solved using IPOPT and SQP. A comparison of runtime is shown in Fig. 4. All simulations were executed on an Intel i5-6200U CPU with 12 GB RAM machine running Ubuntu 14.04. PANOC exhibits very low runtime and outperforms all other solvers by approximately two orders of magnitude.

V. Experimental results and discussion

The proposed NMPC algorithm is validated on an in-house mobile robot with a trailer (Fig. 2). The robot has four independent DC motors with Mecanum wheels, which render it holonomic. A low-level microcontroller implements a velocity controller for each motor. Therefore, from a high-level perspective, the robot can be treated as a velocity-steered holonomic device. A trailer is attached to the robot and the angle between robot and trailer is measured with a rotary potentiometer. A ceiling camera detects the robot's absolute

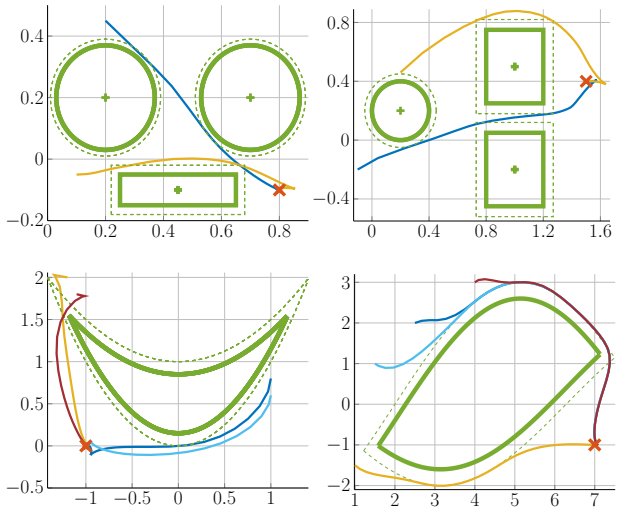

Fig. 3. Four obstacle avoidance scenarios using PANOC. The red x mark denotes the destination point and various lines are trajectories from different initial points. The dashed lines correspond to enlargements of obstacles which are circumscribed by thick green lines. (First row) Obstacle avoidance with circular and rectangular obstacles, (Down, left) Enlarged obstacle defined by $O = \{(x, y) : y > x^2, y < 1 + x^2/2\}$ and (Down, right) Enlarged obstacle defined by $O = \{(x, y) : y > 2\,\mathbf{sin}(-x/2), y < 3\,\mathbf{sin}(x/2 - 1), 1{<}x{<}8\}$.

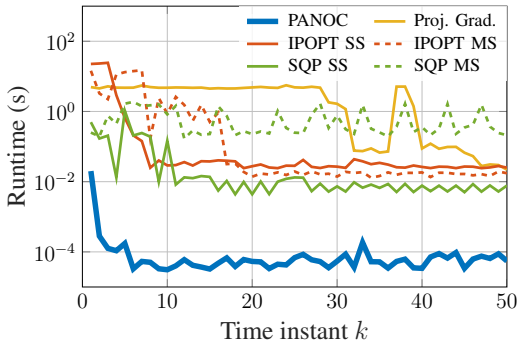

Fig. 4. Comparison of runtime to solve the NMPC problem for four solvers: PANOC, IPOPT and `fmincon`'s SQP algorithm (for the single and multiple shooting formulations) and projected gradient (ForBES implementation). These timings correspond to the navigation problem presented in Fig. 3 (upper right subfigure), starting from the initial point $x_0 = (-0.1, -0.2, \pi/5)$ and with $N = 50$. The tolerance was set to $3 \cdot 10^{-3}$ for all solvers. All solvers are warm-started with their previous solution.

position and orientation using a marker attached on top of the robot. This information is sent to the robot over Wi-Fi and is merged with local encoder measurements to retrieve a fast and accurate estimate of its pose. An Odroid XU4 platform runs the NMPC algorithm on board the robotic platform. PANOC is implemented in C following the C89 standard without external dependencies and can, therefore, be readily executed on embedded systems. The C code for performing Alg. 1 was generated using the AD tool CasADi [25].

The NMPC algorithm is used to steer the vehicle to a desired destination and trailer orientation $x_{\mathrm{ref}} = (3.77, 1.40, 0.0)$. The system dynamics is discretized using the Euler method. A quadratic cost function is employed with $Q_k = Q_N = 0.1 \cdot I_3$ and $R_k = 0.01 \cdot I_2$. Box constraints are imposed on the inputs with $u_{\min} = -0.8\,\mathrm{m/s}$ and $u_{\max} = 0.8\,\mathrm{m/s}$. The obstacles are modelled as discussed in Section II-C and are slightly enlarged for safety so as to

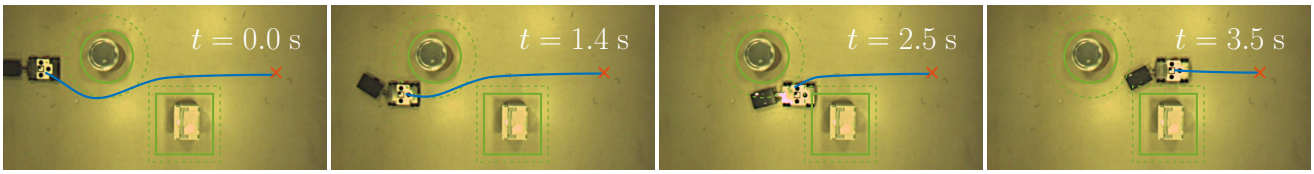

Fig. 5. Point-to-point motion of a holonomic robot with a trailer. The robot avoids a circular and rectangular obstacle indicated with the green lines. The predicted position trajectories are represented in blue and the target position is indicated with the red cross.

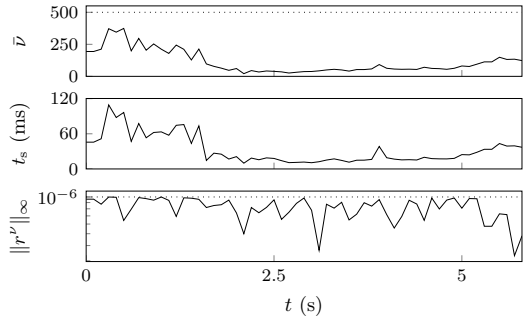

Fig. 6. Number of iterations $\bar{\nu}$, solve time t_s and final residual $\|r^\nu\|_\infty$ for each NMPC cycle.

completely avoid collisions. A horizon length of $N = 50$ is used and the NMPC control rate is 10 Hz. Fig. 5 shows the resulting motion of the robot and the predicted position trajectories at four time instants. PANOC is terminated when $\|r^\nu\|_\infty \leq 10^{-6}$ or if the number of iterations reaches 500. The number of iterations $\bar{\nu}$, the solving time t_s on the Odroid XU4 and the norm of the fixed point residual at termination at each time instant k are shown in Fig. 6.

Additional material and videos from the experiments are found at https://kul-forbes.github.io/PANOC.

VI. Conclusions and Research Directions

We proposed a novel framework which enables us to encode nonconvex obstacle avoidance constraints as a smooth nonlinear equality constraint. This offers a flexible modeling framework and allows the formulation of nonlinear MPC problems with obstacles of complex nonconvex geometry. The resulting MPC problem is solved with PANOC which has favorable theoretical convergence properties and outperforms state-of-the-art NLP solvers. This framework was tested using a C89 implementation of PANOC on a lab-scale system.

Future work will focus on the development of a proximal Lagrangian framework for the online adaptation of the weight parameters in the obstacle avoidance penalty functions — cf. (12) — so that (predicted) constraint violations, modeled by $\tilde{h}(z_k)$, are below a desired tolerance. Moreover, the use of semismooth Newton directions in PANOC will lead to quadratic convergence and superior performance [21].

References

[1] O. Takahashi and R. J. Schilling, "Motion planning in a plane using generalized voronoi diagrams," *IEEE Trans. Rob. Autom.*, vol. 5, no. 2, pp. 143–150, Apr 1989.

[2] J. Minguez, F. Lamiraux, and J.-P. Laumond, *Motion Planning and Obstacle Avoidance*. Cham: Springer, 2016, pp. 1177–1202.

[4] P. Wang and B. Ding, "A synthesis approach of distributed model predictive control for homogeneous multi-agent system with collision avoidance," *Int. J. Control*, vol. 87, no. 1, pp. 52–63, 2014.

[3] D. Bareiss and J. van den Berg, "Generalized reciprocal collision avoidance," *Int. J. Robot. Res.*, vol. 34, no. 12, pp. 1501–1514, 2015.

[5] T. Mercy, R. V. Parys, and G. Pipeleers, "Spline-based motion planning for autonomous guided vehicles in a dynamic environment," *IEEE Trans. Control Syst. Technol.*, vol. PP, no. 99, pp. 1–8, 2017.

[6] F. Debrouwere, W. Van Loock, G. Pipeleers, M. Diehl, J. De Schutter, and J. Swevers, "Time-optimal path following for robots with object collision avoidance using Lagrangian duality," in *9th IEEE Int. Workshop Rob. Mot. Conntrol (RoMoCo)*, 2013, pp. 186–191.

[7] B. Alrifaee, M. G. Mamaghani, and D. Abel, "Centralized non-convex model predictive control for cooperative collision avoidance of networked vehicles," in *IEEE ISIC*, Oct 2014, pp. 1583–1588.

[8] J. V. Frasch, A. Gray, M. Zanon, H. J. Ferreau, S. Sager, F. Borrelli, and M. Diehl, "An auto-generated nonlinear MPC algorithm for real-time obstacle avoidance of ground vehicles," in *Eur. Control Conf.*, 2013, pp. 4136–4141.

[9] V. Turri, A. Carvalho, H. E. Tseng, K. H. Johansson, and F. Borrelli, "Linear model predictive control for lane keeping and obstacle avoidance on low curvature roads," in *IEEE Intell. Transp. Syst. Conf.*, Oct 2013, pp. 378–383.

[10] J. Nocedal and S. J. Wright, *Numerical Optimization*. Springer New York, 2006.

[11] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[12] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathem. Progr.*, vol. 106, no. 1, pp. 25–57, Mar 2006.

[13] M. Diehl, H. G. Bock, and J. P. Schlöder, "A real-time iteration scheme for nonlinear optimization in optimal feedback control," *SIAM J. Contr. Optim.*, vol. 43, no. 5, pp. 1714–1736, 2005.

[14] H. Attouch, J. Bolte, and B. F. Svaiter, "Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward–backward splitting, and regularized Gauss–Seidel methods," *Math. Progr.*, vol. 137, no. 1, pp. 91–129, Feb 2013.

[15] D. P. Bertsekas, *Nonlinear programming*. Athena Scientific, 1999.

[16] L. Stella, A. Themelis, P. Sopasakis, and P. Patrinos, "A simple and efficient algorithm for nonlinear model predictive control," in *IEEE CDC*, Melbourne, Australia, 2017.

[17] E. Gilbert and D. Johnson, "Distance functions and their application to robot path planning in the presence of obstacles," *IEEE J. Rob. Autom.*, vol. 1, no. 1, pp. 21–30, Mar 1985.

[18] H. Scheel and S. Scholtes, "Mathematical programs with complementarity constraints: Stationarity, optimality, and sensitivity," *Math. Op. Res.*, vol. 25, no. 1, pp. 1–22, 2000.

[19] Y.-C. Liang and G.-H. Lin, "Stationarity conditions and their reformulations for mathematical programs with vertical complementarity constraints," *J. Optim. Theory Appl.*, vol. 154, no. 1, pp. 54–70, 2012.

[20] J. C. Dunn and D. P. Bertsekas, "Efficient dynamic programming implementations of Newton's method for unconstrained optimal control problems," *J. Optim. Theory & Appl.*, vol. 63, no. 1, pp. 23–38, 1989.

[21] P. Patrinos, L. Stella, and A. Bemporad, "Forward-backward truncated Newton methods for convex composite optimization," *arXiv:1402.6655*, Feb. 2014.

[22] L. Stella, A. Themelis, and P. Patrinos, "Forward-backward quasi-Newton methods for nonsmooth optimization problems," *Comput. Optim. Appl.*, vol. 67, no. 3, pp. 443–487, Jul 2017.

[23] A. Themelis, L. Stella, and P. Patrinos, "Forward-backward envelope for the sum of two nonconvex functions: Further properties and nonmonotone line-search algorithms," *ArXiv preprint*, jun 2016.

[24] P. Patrinos and A. Bemporad, "Proximal Newton methods for convex composite optimization," in *IEEE CDC*, Dec 2013, pp. 2358–2363.

[25] J. Andersson, "A general-purpose software framework for dynamic optimization," PhD thesis, KU Leuven, Dept. Electr. Eng. (ESAT/SCD) & Optimization in Engineering Center, October 2013.